

Real-Time Computation at the Edge of Chaos in Recurrent Neural Networks

Nils Bertschinger

Institute for Theoretical Computer Science
Technische Universität Graz
A-8010 Graz, Austria
nilsb@igi.tugraz.at

Thomas Natschläger

Software Competence Center Hagenberg
A-4232 Hagenberg, Austria
Thomas.Natschlaeger@scch.at

Abstract

Depending on the connectivity recurrent networks of simple computational units can show very different types of dynamics ranging from totally ordered to chaotic. We analyze how the type of dynamics (ordered or chaotic) exhibited by randomly connected networks of threshold gates driven by a time varying input signal depends on the parameters describing the distribution of the connectivity matrix. In particular we calculate the critical boundary in parameter space where the transition from ordered to chaotic dynamics takes place. Employing a recently developed framework for analyzing real-time computations we show that only near the critical boundary such networks can perform complex computations on time series. Hence, this result strongly supports conjectures that dynamical systems which are capable of doing complex computational tasks should operate near the edge of chaos, i.e. the transition from ordered to chaotic dynamics.

1 Introduction

A central issue in the field of computer science is to analyze the computational power of a given system. Unfortunately there exists no full agreement regarding the meaning of abstract concepts such as *computation* and *computational power*. In the following we mean with a computation simply an algorithm, system or circuit that assigns outputs to inputs. The computational power of a given system can be assessed by evaluating the complexity and diversity of associations of inputs to outputs that can be implemented on it.

Most work regarding the computational power of neural networks is devoted to the analysis of artificially constructed networks; see e.g. (Siu et al., 1995). Much less is known about the computational capabilities¹ of networks with a biologically more adequate connectivity structure. Often randomly connected (with regard to a particular distribution) networks are considered as a suitable model for biological neural systems; see e.g. (van Vreeswijk. and Sompolinsky, 1996). Obviously not only the dynamics of such random networks depends on the connectivity structure but also its computational capabilities. Hence it is of particular interest to analyze the relationship between the dynamical behavior of a system and its computational capabilities.

It has been proposed that extensive computational capabilities are achieved by systems whose dynamics is neither chaotic nor ordered but somewhere in-between order and chaos. This has led to the idea of “*computation at the edge of chaos*”. Early evidence for this hypothesis has been reported by Langton (Langton, 1990) and Packard (Packard, 1988). Langton based his conclusion on data gathered from a parameterized survey of cellular automaton behavior. Packard, on the other hand, used a genetic algorithm to evolve a particular class of complex rules. He found that as evolution proceeded the population of rules tended to cluster near the critical region — i.e. the region between order and chaos — identified by Langton. Similar experiments involving Boolean networks have been conducted by Kauffman (Kauffman, 1993). The main focus of this experiments was to determine the transition from ordered to chaotic behavior in the space of a few parameters controlling the network structure. In fact, the results of numerous numerical simulations suggested that there is a sharp transition between the ordered and chaotic regime. Later on this results were confirmed theoretically by Derrida and others (Derrida and Pomeau, 1986; Derrida and Weisbuch, 1986). They used ideas from statistical physics to develop an accurate mean-field theory for random Boolean networks (Derrida and Pomeau, 1986; Derrida and Weisbuch, 1986) and for Ising-spin models (networks of threshold elements) (Derrida, 1987) which allows to determine the critical parameters analytically.

Because of the physical background this theory focused on the autonomous dynamics of the system, i.e. its relaxation from an initial state (the input) to some terminal state (the output) without any external influences. However such “offline” computations (whose main inspiration comes of course from its similarity

¹Usually one talks only about the computational power of a dedicated computational model which we do not yet have defined. So we prefer the term computational capabilities in combination with a randomly constructed system instead.

to the batch processing by Turing machines) do not adequately describe the input to output relation of systems like animals or autonomous robots which must react in real-time to a continuously changing stream of sensory input. Such computations are more adequately described as mappings, also called filters, from a time varying input signal to a time varying output signal. In this article we will focus on such time-series computations which shifts the emphasis towards online computations and anytime algorithms (i.e., algorithms that output at any time the currently best guess of an appropriate response), and away from offline computations.

The purpose of this paper is to investigate how the computational capabilities of randomly connected recurrent neural networks in the domain of real-time processing and the type of dynamics of the network are related to each other. In particular we will develop answers to the following questions:

- (A) Are there general dynamical properties of input-driven recurrent networks which support a large diversity of complex real-time computations, and if yes,
- (B) can such recurrent networks be utilized to build arbitrary complex filters, i.e. mappings from input time-series to output time-series.

A similar study using leaky integrate-and-fire neurons, which was purely based on computer simulations, was conducted in (Maass et al., 2002).

The rest of the paper is organized as follows. After defining the network model in section 2 we will extend the mean-field theory developed in (Derrida, 1987) to the here considered case of input driven networks. In particular we will determine in section 3 where the transition from ordered to chaotic dynamics occurs in the space of a few connectivity parameters. In section 4 we propose a new complexity measure which can be calculated using the mean-field theory developed in section 3 and serves as a predictor for the computational capability. Finally we investigate in section 5 the relationship between network dynamics and the computational capabilities in the time-series domain.

2 Network Dynamics

To analyze real-time computations on time-series in recurrent neural networks we consider *input driven* recurrent networks consisting of N threshold gates with states $x_i \in \{-1, +1\}$, $i = 1, \dots, N$. Each node i receives nonzero incoming

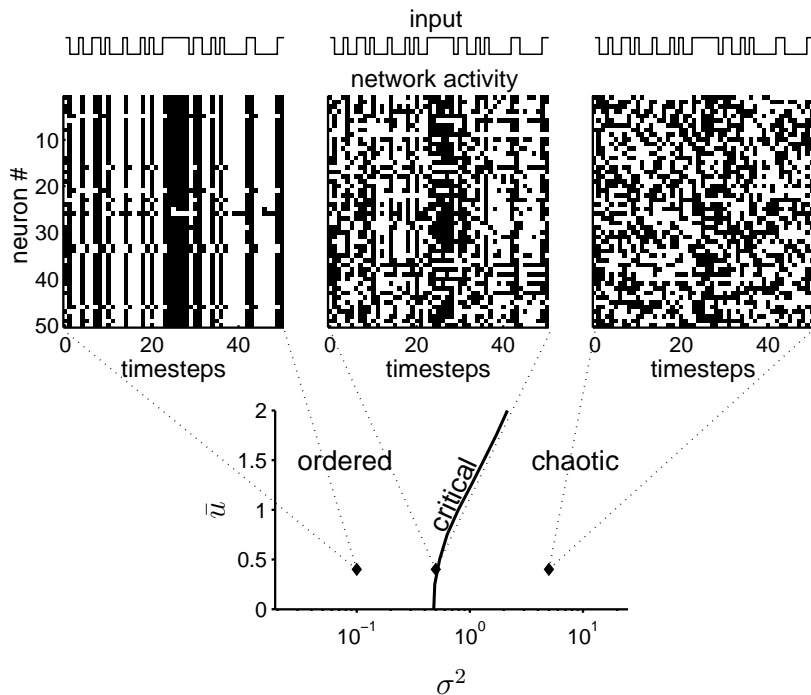


Figure 1: Networks of randomly connected threshold gates can exhibit ordered, critical and chaotic dynamics (see text for a definition of order and chaos). In the upper row examples of the time evolution of the network state $\mathbf{x}(t)$ are shown (black: $x_i(t) = +1$, white: $x_i(t) = -1$) for three different networks with parameters taken from the ordered, critical and chaotic regime respectively. Parameters: $K = 4$, $N = 250$ and σ^2 and \bar{u} as indicated in the phase plot below.

weights w_{ij} from exactly K randomly chosen units $j \in \{1, \dots, N\}$. Such a nonzero connection weight w_{ij} is randomly drawn from a Gaussian distribution with zero mean and variance σ^2 . Furthermore the network is driven by an external input signal $u(\cdot)$ which is applied to each threshold gate. Hence, in summary the update of the network state $\mathbf{x}(t) = (x_1(t), \dots, x_N(t))$ is given by

$$x_i(t) = \Theta \left(\sum_{j=1}^N w_{ij} \cdot x_j(t-1) + u(t) \right) \quad (1)$$

which is applied for all neurons $i \in \{1, \dots, N\}$ in parallel and where $\Theta(h) = +1$ if $h \geq 0$ and $\Theta(h) = -1$ otherwise.

In the following we consider a randomly drawn binary input signal $u(\cdot)$: at each time step $u(t)$ assumes the value $\bar{u} + 1$ with probability r and the value $\bar{u} - 1$ with probability $1 - r$. Note that the input bias \bar{u} can also be considered as the mean threshold of the nodes.

Similarly to autonomous systems these networks can also exhibit very different types of dynamics ranging from completely ordered all the way to chaotic depending on the connectivity structure. Informally speaking we will call a network chaotic if arbitrary small differences in a (initial) network state $\mathbf{x}(0)$ are highly amplified and do not vanish. In contrast a totally ordered network forgets immediately about a (initial) network state $\mathbf{x}(0)$ and the current network state $\mathbf{x}(t)$ is determined to a large extent by the current input $u(t)$ (see section 3 for a precise definition of order and chaos).

The top row of Fig. 1 shows typical examples of ordered, critical and chaotic dynamics while it is indicated in the lower panel (*phase plot*) which values of the model or system parameters correspond to a given type of dynamics. We will concentrate on the three system parameters K , σ^2 , and \bar{u} where K (number of incoming connections per neuron) and σ^2 (variance of nonzero weights) determine the connectivity structure whereas \bar{u} describes a statistical property (the mean or bias) of the input signal $u(\cdot)$. A *phase transition* occurs if varying such a parameter leads to a qualitative change in its behavior like switching from ordered to chaotic dynamics. An example can be seen in Fig. 1 where increasing the variance σ^2 of the weights leads to chaotic behavior. We refer to the transition from the ordered to the chaotic regime as the *critical line*.

3 Finding the critical line

To define the chaotic and the ordered phase of a discrete dynamical system Derrida and Pomeau (Derrida and Pomeau, 1986) proposed the following approach: consider two (initial) network states with a certain (normalized) Hamming distance. These states are mapped to their corresponding following states defined by the network dynamics and the change in the Hamming distance is observed. If the distances tend to grow this is a sign of chaos whereas if the distance decreases this is a signature of order (Derrida and Pomeau, 1986; Derrida and Weisbuch, 1986; Derrida, 1987). This is closely related to the computation of the Lyapunov exponent in a continuous system (see e.g. (Luque and Sole, 2000)) since this approach also emphasizes the idea to consider a system as chaotic if it is very

sensitive to differences in its initial conditions. We will apply the same basic approach to define the chaotic and ordered regime for an *input driven* network: Two (initial) states are mapped to their corresponding following states with the same input in each case while the change in the Hamming distance is observed. Again if the distances tend to increase (decrease) this is a sign for chaos (order).

Following closely the arguments by Derrida (Derrida, 1987) we develop a mean-field theory (i.e. we consider the limit $N \rightarrow \infty$) which allows to analyze the time evolution of the (normalized) Hamming distance $d(t) = |\{i : x_{1,i}(t) \neq x_{2,i}(t)\}|/N$ between two network states $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ where $u(\cdot)$ is the input signal ($x_{l,i}(t)$ denotes the i -th component of the network state vector $\mathbf{x}_l(t)$, $l = 1, 2$). As it is shown in appendix A the Hamming distance $d(t)$ between two trajectories at time t is related to the Hamming distance $d(t+1; u)$ at time $t+1$ (given that u is the input in that time-step) through the equation

$$d(t+1; u) = \sum_{c=0}^K \binom{K}{c} \cdot d(t)^c \cdot (1-d(t))^{K-c} \cdot P_{BF}(c, u) \quad (2)$$

where $P_{BF}(c, u)$ is the probability that exactly c bit-flips out of the K inputs to a unit will cause a different output of that unit in the two trajectories (see appendix A for details).²

In contrast to previous studies (Derrida, 1987; Derrida and Weisbuch, 1986) the Hamming distance $d(t+1; u)$ in Equ. (2) depends on the input u which leads to the new states at time $t+1$. If one assumes a certain distribution of the input signal one can calculate the expected value of $d(t+1; u)$. Let us consider the case where $u = \bar{u} + 1$ and $u = \bar{u} - 1$ with probability r and $1-r$ respectively. This input distribution leads to the equation

$$d_{fade}(t+1) := \text{FADE}(d(t)) := r \cdot d(t+1; \bar{u} + 1) + (1-r) \cdot d(t+1; \bar{u} - 1) \quad (3)$$

for the time evolution of the Hamming distance.

The map defined by Equ. (3) determines whether a network is in the ordered phase or not: If for all initial values $d(0) \in [0, 1]$ the Hamming distance $d_{fade}(t)$ converges to zero for $t \rightarrow \infty$ we know that any state differences will die out and the network is in the ordered phase. If on the other hand a small difference $d(t) > 0$

²Note that in the appendix $d(t+1; u)$ and $P_{BF}(c, u)$ are denoted as $d(t+1; u, u)$ and $P_{BF}(c, u, u)$ respectively.

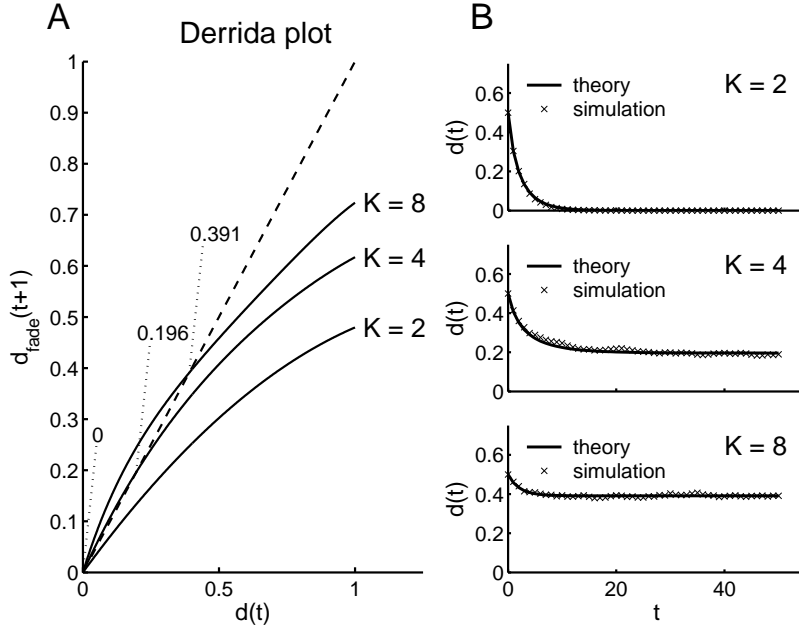


Figure 2: **A** Derrida plots. The Derrida plots show the dependence of $d_{fade}(t+1)$ on $d(t)$. Shown are Derrida plots for networks with parameters $\bar{u} = 0$, $\sigma^2 = 1$, $r = 0.5$ and three different values of K as denoted in the figure. For each plot the numerical value of the stable fixed-point $d_{fade}^* = \text{FADE}(d_{fade}^*)$, i.e. the intersection with the identity function (dashed line), is given. **B** Comparison of the mean field theory and numerical simulations. For each of the considered values of K we compare the theoretical evolution of the Hamming distance by iterating Equ. (3) (solid line) to results from numerical simulations (crosses). Simulation results are averages over 50 runs of the same network of $N = 250$ threshold gates where in each run the input $u(\cdot)$ was randomly generated with “rate” $r = \Pr \{u(t) = \bar{u} + 1\}$.

is amplified and never dies out the network is in the chaotic phase. Whether the Hamming distance will converge to zero or not can easily be seen in a so called Derrida-Plot where $d_{fade}(t+1) = \text{FADE}(d(t))$ is plotted versus $d(t)$.

A network is in the ordered phase iff $d_{fade}^* = 0$ is the only stable fixed-point $d_{fade}^* = \text{FADE}(d_{fade}^*)$ of Equ. (3). See Fig. 2A for examples of Derrida Plots.³ We

³Equ. (3) was solved numerically to yield the Derrida plots.

also compared the theoretical predictions of Equ. (3) to simulations of networks. The results are shown in Fig. 2B. Even so the theory assumes an infinite network size and that the weights are randomly drawn at each time step (the annealing approximation (Derrida and Pomeau, 1986)) a good correspondence with simulation results is already obtained for networks containing a few hundred nodes and with the weights randomly drawn prior to the simulation (see appendix A for a more detailed discussion about the annealing approximation).

The stability can be checked by analyzing the slope α of the Derrida plot at $d(t) = 0$. The fixed-point $d_{fade}^* = 0$ is stable if $|\alpha| < 1$. As shown in appendix B the slope α is given as

$$\alpha = \left. \frac{\partial d_{fade}(t+1)}{\partial d(t)} \right|_{d(t)=0} = K \cdot (r \cdot P_{BF}(1, \bar{u} + 1) + (1 - r) \cdot P_{BF}(1, \bar{u} - 1))$$

Hence the stability of the fixed-point $d_{fade}^* = 0$ depends only on the probability $P_{BF}(1, \cdot)$ that a single changed state component leads to a change of the output. Therefore the so called critical line $|\alpha| = 1$ where the phase transition from ordered to chaotic behavior occurs is given by

$$r \cdot P_{BF}(1, \bar{u} + 1) + (1 - r) \cdot P_{BF}(1, \bar{u} - 1) = \frac{1}{K} \quad (4)$$

This criterion is related to criticality criteria that are derived using a single bit-flip approximation directly (Rohlf and Bornholdt, 2002).

The corresponding phase diagram is shown in Fig. 3. For fixed values of \bar{u} and σ^2 more incoming connections K tend to make the network more chaotic. Note that for $K = 1$ and $K = 2$ the network can not be made chaotic (see appendix B for details). Ordered dynamics can be achieved by means of smaller internal connection weights (smaller σ^2) which will increase the impact of the input signal or by biasing the network towards -1 or +1 by changing the mean \bar{u} of the input signal. This second way to achieve ordered dynamics is analogous to using biased Boolean functions in random Boolean networks (Kauffman, 1993). Since the state encoding of the networks is symmetric using a positive or negative mean \bar{u} is equivalent. Usually a higher bias is expected to give more ordered dynamics. However, for high values of K this is not true for \bar{u} near ± 1 which is an effect due to the binary input signals that are used here.

Note that the ordered phase can be described by using the notion of *fading memory* (Boyd and Chua, 1985). Informally a network has fading memory if the current

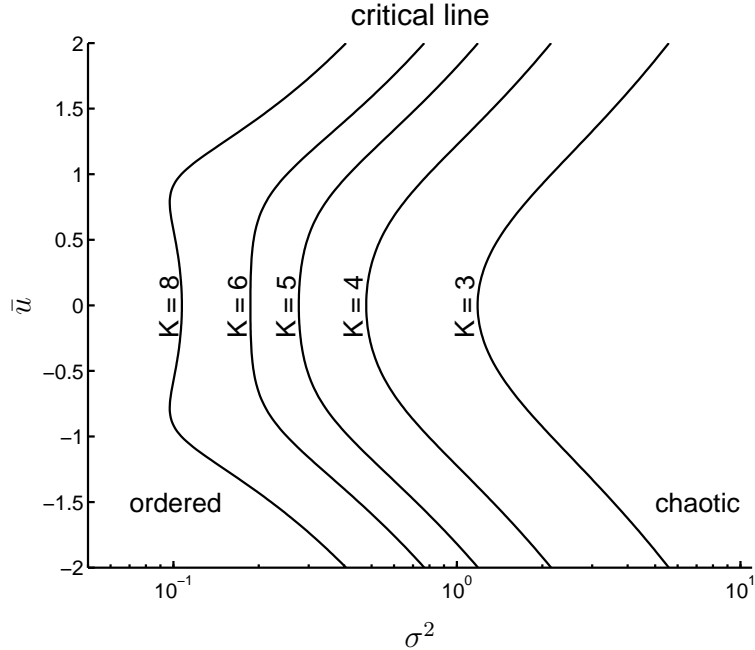


Figure 3: The critical line. Shown is the critical line for parameters σ^2 and \bar{u} in dependence of K (denoted in the figure). As before we set $r = 0.5$. Note that the ordered (chaotic) regime is to the left (right) of the critical line.

state $\mathbf{x}(t)$ depends just on the values of its input $u(t')$ from some finite time window $t' \in [t - T, t]$ into the past⁴(Maass et al., 2002). A slight reformulation of this property (Jaeger, 2002) shows that it is equivalent to the requirement that all state differences vanish, i.e. being in the ordered phase. This property is called “echo state property” in (Jaeger, 2002).

Fading memory plays an important role in the “liquid state machine” framework (Maass et al., 2002) (called “echo state networks” in (Jaeger, 2002)) which we will

⁴More formally a network is said to have the fading memory property if the following holds: For all $\epsilon > 0$, initial conditions $\mathbf{x}_1(0), \mathbf{x}_2(0)$ and input signals $u_1(\cdot), u_2(\cdot)$ exist $\delta > 0$ and $T \in \mathbb{N}$ such that $\|u_1 - u_2\|_{input} < \delta \Rightarrow \|\mathbf{x}_1(T) - \mathbf{x}_2(T)\|_{state} < \epsilon$. Here $\mathbf{x}_1(T), \mathbf{x}_2(T)$ denote the states that are obtained by running the network T time-steps on initial conditions $\mathbf{x}_1(0), \mathbf{x}_2(0)$ and input signals $u_1(\cdot), u_2(\cdot)$ respectively. $\|\cdot\|_{input}$ and $\|\cdot\|_{state}$ are norms that turn the space of input signals and network states into compact vector spaces.

employ (and discuss in more detail) in section 5 to show that networks operating near the critical line support complex computations. Intuitively speaking in a network with fading memory a state $x(t)$ is fully determined by a finite history $u(t-T), u(t-T+1), \dots, u(t-1), u(t)$ of the input $u(\cdot)$.⁵ This would in principle allow an appropriate *readout function* to deduce the recent input, or any function of it, from the network state. If on the other hand the network does not have fading memory (i.e. is in the chaotic regime) a given network state $x(t)$ also contains “spurious” information about the initial conditions and hence it is hard or even impossible to deduce any features of the recent input.

4 Separation as a predictor for computational power

A further dynamical property that is especially important if a network is to be useful for computations on input time-series is the *separation property* (see also (Maass et al., 2002)): any two different input time series which should produce different outputs (with regard to the readout function) drive the recurrent network into two (significantly) different states. From the point of view of the readout function it is clear that different outputs can only be produced for different network states. Hence only if different input signals separate the network state (i.e. different inputs result in different states) it is possible for a readout function to respond differently. Furthermore it is desirable that the separation (distance between network states) increases with the difference of the input signals.

However, a simple separation measurement can not be directly related to the computational power since chaotic networks separate even minor differences in the input to a very high degree. This is because any single different bit of the network state (eventually caused by the differing inputs) results in drastically different trajectories but is not a direct consequence of the different inputs. Therefore we will propose in this section the so called “network mediated” separation (*NM*-separation for short) which aims to capture those portions of the separation which are due to the differences in the input signals $u_1(\cdot)$ and $u_2(\cdot)$.

As before we measure differences of network states by their Hamming distance. The distance of input signals $u_1(\cdot)$ and $u_2(\cdot)$ is defined by $b =$

⁵More precisely there exists a function E such that $x(t) = E(u(t-T), u(t-T+1), \dots, u(t-1), u(t))$ for some finite $T \in \mathbb{N}$.

$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^T (1 - \delta(u_1(t), u_2(t)))$ which measures the average number of differences in the input signals, where $\delta(x, y) = 1$ if $x = y$ and $\delta(x, y) = 0$ otherwise. b can also be interpreted as the probability that $u_1(t)$ and $u_2(t)$ are different, i.e. $b = \Pr \{u_1(t) \neq u_2(t)\}$.

The mean field theory we have developed (see appendix A) can easily be extended to calculate the state distance (separation) that results from applying inputs $u_1(\cdot)$ and $u_2(\cdot)$ with a mean distance of b (r denotes the ‘‘rate’’ $r = \Pr \{u_1(t) = \bar{u} + 1\}$ of $u_1(\cdot)$):

$$d_{sep}(t+1) = \text{SEP}(d(t)) \tag{5}$$

$$\begin{aligned} &= b \cdot (r \cdot d(t+1; \bar{u} + 1, \bar{u} - 1) + (1 - r) \cdot d(t+1; \bar{u} - 1, \bar{u} + 1)) \\ &\quad + (1 - b) \cdot (r \cdot d(t+1; \bar{u} + 1, \bar{u} + 1) + (1 - r) \cdot d(t+1; \bar{u} - 1, \bar{u} - 1)) \end{aligned} \tag{6}$$

As in Sec. 3 we numerically solved Equ. (6) for the stable fixed-point $d_{sep}^* = \text{SEP}(d_{sep}^*)$, i.e. the separation that results from long runs with inputs $u_1(\cdot)$ and $u_2(\cdot)$ with a distance of b .

The part of this separation that is caused by the input distance b and not by the distance of some initial state is then given by $d_{sep}^* - d_{fade}^*$ since d_{fade}^* measures the state distance that is caused by differences in the initial states and remains even after long runs with the same inputs (see Sec. 3). Note that d_{fade}^* is always zero in the ordered phase and non-zero in the chaotic phase.

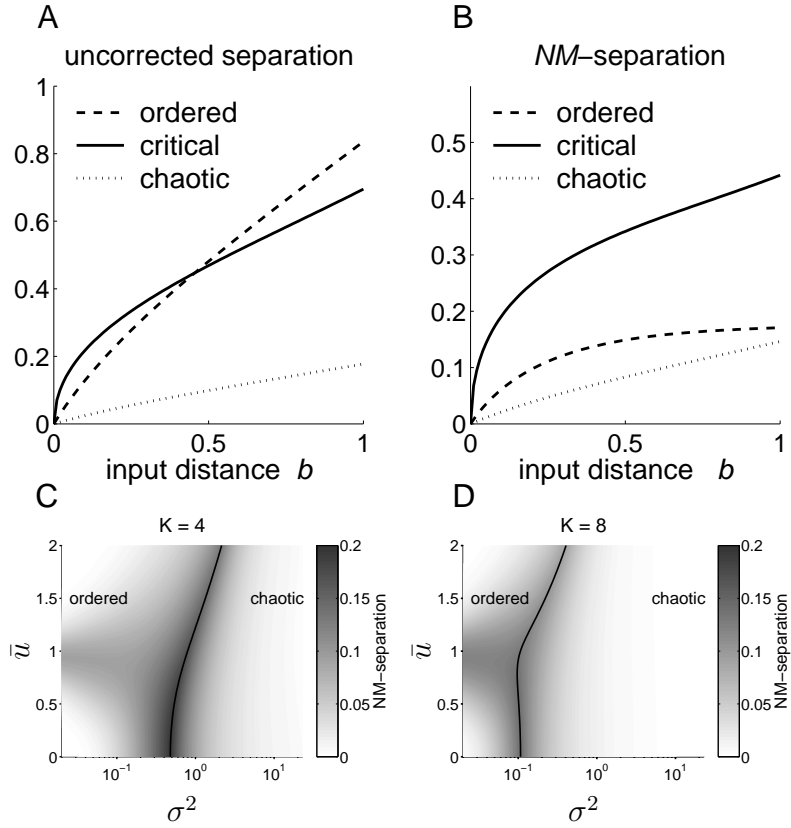


Figure 4: NM -separation assumes high values on the critical line. **A** The uncorrected separation $d_{sep}^* - d_{fade}^*$ is shown for ordered ($\sigma^2 = 0.1$), critical ($\sigma^2 = 0.5$), and chaotic ($\sigma^2 = 5$) dynamics in dependence of the input distance b ($K = 4$, $\bar{u} = 0.4$, $r = 0.5$). **B** Same as panel A but for the NM -separation. **C** The gray coded image shows the NM -separation in dependence on σ^2 and \bar{u} for $K = 4$, $r = 0.5$, and $b = 0.1$ (white: low values, dark gray: high values). The solid line marks the critical line. **D** same as C but for $K = 8$.

Furthermore we also have to consider the immediate separation d_{inp} that is expected from the given input distribution (determined by \bar{u} and b). This term can be estimated as

$$d_{inp} := b \cdot (2 \cdot q(\bar{u}, r) - 1)^2$$

where $q(\bar{u}, r)$ is the fraction of nodes that “copy the input”⁶ (see appendix C for details). The immediate separation d_{inp} is especially high for input driven networks (low values of σ^2 and values of $q(\bar{u}, r)$ close to 1.0). Obviously this immediate separation can not be utilized by any readout function which needs access to information about the input a few time steps ago because the network state $\mathbf{x}(t)$ is dominated to a very large extent by the current input at time t . Or more informally speaking: input driven networks do not have any memory about recent inputs.

Since we want the NM -separation to be a predictor for computational power we have to correct $d_{sep}^* - d_{fade}^*$ by the immediate separation d_{inp} in order to take the loss of memory of input driven networks into account. Hence a suitable measure of the network mediated separation due to input differences is given by

$$NM_{sep} = d_{sep}^* - d_{fade}^* - b \cdot (2 \cdot q(\bar{u}, r) - 1)^2 \quad (7)$$

In Fig. 4A,B the uncorrected separation $d_{sep}^* - d_{fade}^*$ and the NM -separation are plotted in dependence of the input distance b for three representative parameter settings (same as in Fig. 1). As expected both measures are increasing with b . Note that for the ordered dynamics the uncorrected separation assumes similar (or even higher) values as the uncorrected separation for critical parameters (this is because the network is very much input driven; see above). In contrast the NM -separation which is corrected by the immediate separation d_{inp} is highest for critical parameters.

In Fig. 4C,D the NM -separation resulting from an input difference of $b = 0.1$ is shown in dependence of the network parameters \bar{u} and σ^2 . It can be clearly seen that the separation peaks at the critical line. Because of the computational importance of the separation property this also suggests that the computational capabilities of the networks will peak at the onset of chaos, which is confirmed in the next section.

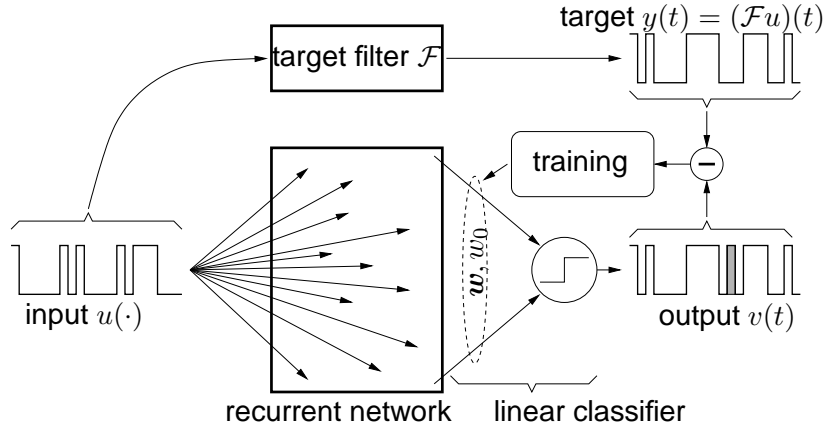


Figure 5: Setup used to train a linear classifier for a given computational task. The input $u(\cdot)$ drives the dynamic reservoir (the randomly connected recurrent network in our case) while a memoryless readout function C which has access to the full network state $\mathbf{x}(t)$ computes the actual output $v(t) = C(\mathbf{x}(t))$. Only the weights $\mathbf{w} \in \mathbb{R}^N$, $w_0 \in \mathbb{R}$ of the linear classifier are adjusted such that the actual network output $v(t) = \Theta(\mathbf{w} \cdot \mathbf{x}(t) + w_0)$ is as close as possible to the target values $y(t) = (\mathcal{F}u)(t)$. The signals shown are an example of the performance of a trained network on a one step delayed XOR task, i.e. $y(t) = \text{XOR}(u(t-1), u(t-2))$.

5 Real time computations at the edge of chaos

To access the computational power of a network we make use of the so called “liquid state machine” framework which was proposed by Maass et.al. (Maass et al., 2002) and independently by Jaeger (Jaeger, 2002) who called it “echo state networks”. They put forward the idea that any complex time-series computation can be implemented by a system which consists of two conceptually different parts: a dynamical system with “rich” dynamics followed by a memoryless readout function (the basic architecture is depicted in Fig. 5). This idea is based on the following observation: If one excites a sufficiently complex recurrent circuit with an input signal, and looks at a later time at the current internal state of the circuit, then this state is likely to hold a substantial amount of information about recent inputs. In order to implement a specific task — called the target filter in the

⁶A unit copies the input if it outputs +1 if the input has the value $\bar{u} + 1$ and outputs -1 if the input has the value $\bar{u} - 1$.

following — it is enough that the readout function is able to extract the relevant information from the network state. This amounts to a classical pattern recognition problem, since the temporal dynamics of the input stream has been transformed by the recurrent circuit into a high dimensional spatial pattern. Following this line of arguments the construction of a liquid state machine which implements an arbitrary filter can be decomposed into two steps: a) choose a proper general-purpose recurrent network and b) train a readout function to map the network state to the desired output.

This approach is potentially successful if the general-purpose network encodes the relevant features of the input signal in the network state in such a way that the readout function can easily extract it. In the context of the network model considered in this paper we have already investigated in depth two properties which support such a requirement: fading memory (i.e. ordered dynamics) and (network mediated) separation. However it is not yet clear whether these properties are enough to ensure that the information can be extracted “easily”. In fact, up to now it can only be proven (Maass et al., 2002) that any time invariant filter which has fading memory can be approximated with any desired degree of precision if i) the recurrent network has fading memory, ii) the recurrent network has the separation property, and iii) the readout has the capability to approximate any given continuous function that maps current states on current outputs. Hence in a worst case scenario it may happen that the readout function must be arbitrarily complex to implement the desired target filter.

However we will show that near the critical line the network encodes the information in such a way that a simple linear classifier $C(\mathbf{x}(t)) = \Theta(\mathbf{w} \cdot \mathbf{x}(t) + w_0)$ suffices to implement a broad range of complex nonlinear functions.⁷

To access the computational power in a principled way networks with different parameters were tested on a delayed 3 bit parity task. Here the desired output signal $y_\tau(t)$ is given by $\text{PARITY}(u(t-\tau), u(t-\tau-1), u(t-\tau-2))$ for increasing delays $\tau \geq 0$. The task is quite complex for the networks considered here since the parity function is not linear separable and it requires memory. Hence to achieve good performance it is necessary that a state $\mathbf{x}(t)$ contains information

⁷In order to train the network for a given target filter \mathcal{F} only the parameters $\mathbf{w} \in \mathbb{R}^N$, $w_0 \in \mathbb{R}$ of the linear classifier are adjusted such that the actual network output $v(t)$ is as close as possible to the target values $y(t) = (\mathcal{F}u)(t)$. The weights $\mathbf{w} \in \mathbb{R}^N$, $w_0 \in \mathbb{R}$ are determined using standard linear regression in an off-line fashion.

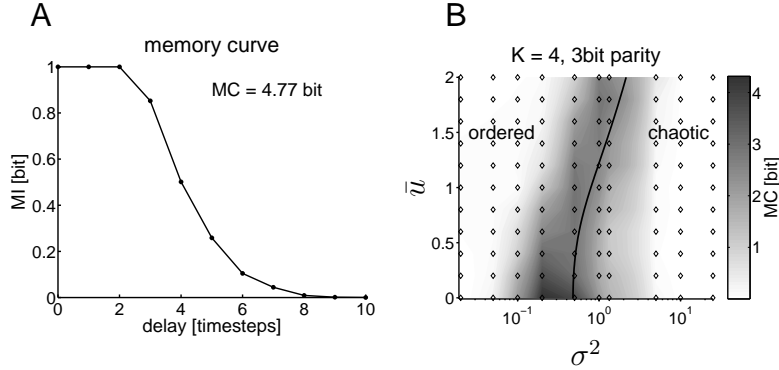


Figure 6: **A** Memory curve for the 3 bit parity task. Shown is the mutual information $MI(v, y)$ between the classifier output $v(\cdot)$ and the target function $y(t) = \text{PARITY}(u(t - \tau), u(t - \tau - 1), u(t - \tau - 2))$ on a test set (see text for details) for various delays τ . Parameters: $N = 250$, $K = 4$, $\sigma^2 = 0.2$, $\bar{u} = 0$ and $r = 0.5$. **B** The gray coded image (an interpolation between the data points marked with open diamonds) shows the performance of trained networks in dependence of the parameters σ^2 and \bar{u} for the same task as in panel A. Performance is measured as the memory capacity $MC = \sum_{\tau} MI(\tau)$, i.e. the “area” under the memory curve. Remaining parameters as in panel A.

about several input bits $u(t')$, $t' < t$ in a nonlinear transformed form such that a linear classifier C is sufficient to perform the nonlinear computation of the parity task.

In Fig. 6A the mutual information $MI(v_{\tau}, y_{\tau})$ ⁸ measured on a test set between the network output $v_{\tau}(\cdot)$ trained on a delay of τ and the target signal $y_{\tau}(\cdot)$ is shown for increasing delays τ (cf. (Natschläger and Maass, 2003)).⁹ Following

⁸The mutual information $MI(v, y)$ (in bits) between two signals $v(\cdot)$ and $y(\cdot)$ is defined as $MI = \sum_{v'} \sum_{y'} p(v', y') \log_2 \frac{p(v', y')}{p(v')p(y')}$ where the sums are over all possible values ($\{-1, +1\}$ in our case) of $v(\cdot)$ and $y(\cdot)$, $p(v', y') = \Pr \{v(t) = v' \wedge y(t) = y'\}$ is the joint probability, $p(v') = \Pr \{v(t) = v'\}$ and $p(y') = \Pr \{y(t) = y'\}$ are the marginal distributions. Note that all these probabilities can reliably be estimated simply by counting since $v(\cdot)$ and $y(\cdot)$ are binary signals in our case.

⁹Note that for each delay τ a separate classifier is trained. For training a single linear classifier a training set $\{\langle x_l, y_l \rangle\}$, $l = 1..9000$ where x_l are network states

(Jaeger, 2002) we define a memory capacity $MC = \sum_{\tau=0}^{\infty} MI(v_{\tau}, y_{\tau})$. In Fig. 6B the dependence of MC on the parameters of the randomly drawn networks is shown. Each data point shows the mean over 10 different networks with the given parameters¹⁰.

The highest performance is clearly achieved for parameter values close to the critical line where the phase transition occurs. This has been noted before (Langton, 1990; Packard, 1988). In (Packard, 1988) for example cellular automata were evolved to solve one particular task. Even so there is evidence that evolution will find solutions near the critical line the specific dynamics selected are highly task specific. For this and other reasons edge of chaos interpretations of these results were criticized (Mitchell et al., 1993). In contrast the networks used here are not optimized for any specific tasks, but only a linear readout function is trained to extract the task specific information that is already present in the state of system. Furthermore each network is evaluated for many different tasks, such as the 3 bit parity of increasingly delayed input values. Therefore a network that is specifically designed for a single task will not show a good performance in this setup. These considerations suggest the following hypotheses regarding the computational function of generic recurrent neural circuits: to serve as general-purpose temporal integrator, and simultaneously as kernel (i.e., nonlinear projection into a higher dimensional space) to facilitate subsequent linear readout of information whenever it is needed.

The network size N determines the dimension of the space into which the input x_l and $y_l \in \{+1, -1\}$ are the target values given by the considered target filter was constructed as follows: The considered network was run from 10 randomly drawn initial states on randomly drawn input signals of length 5000 and rate $r = 0.5$. The first 500 states were discarded and then only the network states of every fifth time-step were added with the according target value y_l to the training set. The weights $w_0 \in \mathbb{R}$, $\mathbf{w} = \langle w_1, \dots, w_N \rangle \in \mathbb{R}^N$ of the linear classifier were then determined by standard linear regression: i.e. by minimizing the squared error $\sum_l (w_0 + \mathbf{w} \cdot \mathbf{x}_l - y_l)^2$ via the pseudo-inverse solution. The performance of the trained classifier was then measured on a test set. Therefore the network was run again from 10 initial states for 2000 time-steps. The initial states and input signals were again randomly drawn and different from the ones used during training. As before the first 500 states were discarded.

¹⁰Standard deviation are not shown because they are quite small (< 0.5 bit of MC). The performance differences across the critical line are therefore highly significant.

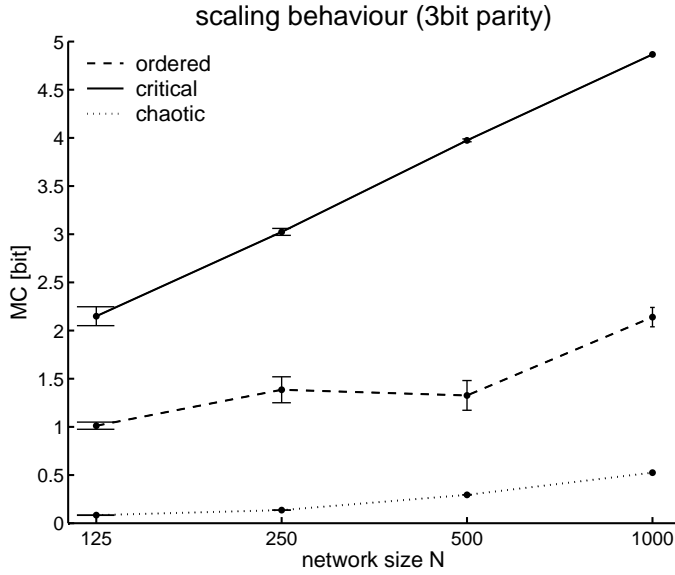


Figure 7: Computational capabilities scale with network size. Shown is the performance on the 3 bit parity task in dependence on the network size N (mean $MC \pm$ standard deviation over 10 randomly drawn networks for each parameter setting). The remaining parameters were chosen as $K = 4, r = 0.5, \bar{u} = 0.4$ and $\sigma^2 = 0.1$ (ordered), $\sigma^2 = 0.5$ (critical) and $\sigma^2 = 5$ (chaotic).

signal is nonlinearly transformed by the network and in which the linear classifier is operating. It is expected that MC increases with N due to the enhanced discrimination power of the readout function. Hence it is worthwhile to investigate how the computational power (in terms of MC) scales up with the network size N ; see Fig. 7. Interestingly the steepest increase of MC with increasing N is observed for critical parameters (almost perfect logarithmic scaling). In contrast at non-critical parameters the performance on the delayed 3 bit parity task grows only slightly with increasing network size.

To further explore the “computation at the edge of chaos” idea the above simulations were repeated for different values of K and different tasks. The networks were trained on a delayed 1 bit (actually just a delay line), a delayed 3 bit and a delayed 5 bit parity task as well as on 50 randomly drawn Boolean functions of the last five inputs, i.e. $y(t) = f(u(t), u(t-1), u(t-2), u(t-3), u(t-4))$ for a randomly drawn Boolean function $f : \{-1, +1\}^5 \rightarrow \{-1, +1\}$. Average results

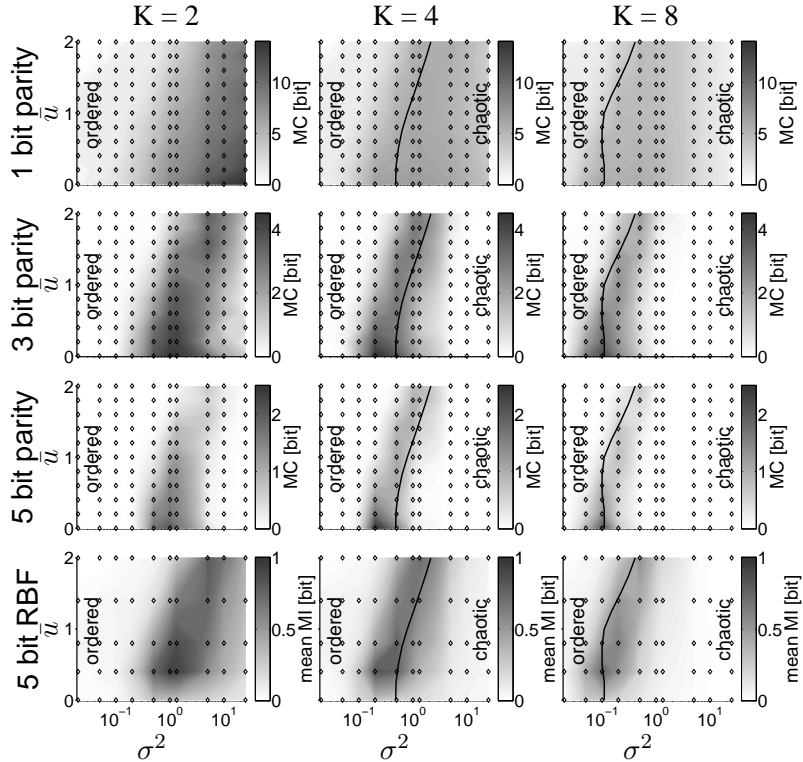


Figure 8: Performance of trained networks for various parameters and different tasks with increasing complexity. The performance (as measured in Fig. 6B) is shown in dependence of the parameters σ^2 and \bar{u} for $K = 2, 4, 8$ (left to right) and the 1, 3, and 5 bit parity task as well as for an average over 50 randomly drawn Boolean functions (top to bottom).

over 10 randomly drawn networks for each parameter setting are shown in Fig. 8. In almost all cases the best performance is achieved close to the critical line. Just for the simplest task considered, a delay line, the best performance is achieved in the chaotic regime. In the case of $K = 2$ the network can never be made chaotic by increasing σ^2 but still the performance peaks at some intermediate value of σ^2 . The reasons for that remain to be uncovered. It is also notable that even so the best performances are usually achieved close to the critical line the performance varies considerably along it. Especially at high values of K the performance drops significantly if the network is stabilized by using large biases \bar{u} . Hence it is advantageous to use unbiased networks (i.e. $\bar{u} = 0$). This is probably related to the fact

that for $\bar{u} = 0$ the entropy of the network states is highest. Normalizing MC on a entropy bound obtained from the mean activity¹¹ suggests that it indeed explains a large amount of the performance drop (not shown). Just for the randomly drawn Boolean functions slightly biased networks show a significantly higher performance, because unbiased networks cannot implement symmetric functions (about half of the randomly drawn functions will be symmetric) since they are antisymmetric devices (compare (Jaeger, 2002)).

6 Discussion

We developed a mean-field theory for input-driven networks which allows an accurate determination of the position of the transition line between ordered and chaotic dynamics with respect to the parameters controlling the network connectivity and input statistics. To our knowledge this is the first time that networks which receive a time varying input signal have been considered in this context. Additionally a clear correlation between the network dynamics and computational power for real-time time-series processing was established. We found that near the critical line the type of networks considered here support a broad class of complex computations. These results provide further evidence for the idea of “computation at the edge of chaos” (Langton, 1990).

In the context of the questions raised in the introduction our findings point to the following answers:

- (A) Dynamics near the critical line are a general property of input driven dynamical systems which support complex real-time computations.
- (B) Such systems can be used by training a simple readout function to approximate any arbitrary complex filter.

Furthermore this suggests that to exhibit sophisticated information processing capabilities an adaptive system should stay close to critical dynamics. Since the dynamics is also influenced by the statistics of the input signal (for example the

¹¹For a given mean activity of $a = \Pr \{x_i(t) = +1\}$ the maximum entropy of a single unit is $H(a) = -a \log a - (1 - a) \log(1 - a)$. Note that $0 \leq H(a) \leq 1$ and $H(0.5) = 1$. Hence an upper bound for the entropy of the network states is $N \cdot H(a)$ which reaches its maximum for $a = 0.5$. To compensate for the entropy drop we considered the scaling $MC \cdot N \cdot H(0.5)/(N \cdot H(a)) = MC/H(a)$.

mean as shown here) these results indicate a new role for plasticity rules: Stabilize the dynamics at the critical line. Such rules would then implement what could be called “dynamical homeostasis” which could be achieved in an unsupervised manner. Examples of such rules that adjust the connectivity K towards the critical line in threshold networks can be found in (Bornhold and Rohlf, 2000) and (Bornholdt and Röhl, 2003). Such a system then actively adjust its dynamics towards criticality, so that it shows self-organized criticality ((Bak et al., 1988; Bak et al., 1987)). The rule described in (Bornholdt and Röhl, 2003) is also interesting from a biological point of view since it is based on local Hebb-like correlations. In the context of artificial neural networks it is shown in (Dauce et al., 1998) that Hebbian learning drives the dynamics from the chaotic into the ordered regime.

Hence combining task-specific optimization provided by (supervised) learning rules (Hertz et al., 1991) with a more general adjustment of the dynamics would allow a system to gather specific information while self-organizing its dynamics towards criticality in order to be able to react flexible to other incoming signals. We believe that this viewpoint will be a fruitful avenue to further explore the mysteries of brain functioning. By combining self-organized criticality with learning it might be possible to uncover some of the principles underlying the most impressive adaptability exhibited by brains.

Acknowledgments

We would like to thank Alexander Kaske for stimulating discussions and helpful comments about the manuscript.

References

- Bak, P., Tang, C., and Wiesenfeld, K. (1987). Self-organized criticality: An explanation of $1/f$ noise. *Physical Review Letters*, 59(4):381–384.
- Bak, P., Tang, C., and Wiesenfeld, K. (1988). Self-organized criticality. *Physical Review A*, 38(1):364–374.
- Bornhold, S. and Rohlf, T. (2000). Topological evolution of dynamical networks: Global criticality from local dynamics. *Physical Review Letters*, 84(26):6114–6117.

- Bornholdt, S. and Röhl, T. (2003). Self-organized critical neural networks. *Physical Review E*, 67:066118.
- Boyd, S. and Chua, L. O. (1985). Fading memory and the problem of approximating nonlinear operators with volterra series. *IEEE Trans. on Circuits and Systems*, 32:1150.
- Dauce, E., Quoy, M., Cessac, B., Doyon, B., and Samuelides, M. (1998). Self-organization and dynamics reduction in recurrent networks: stimulus presentation and learning. *Neural Networks*, 11:521–533.
- Derrida, B. (1987). Dynamical phase transition in non-symmetric spin glasses. *J. Phys. A: Math. Gen.*, 20:721–725.
- Derrida, B. and Pomeau, Y. (1986). Random networks of automata: A simple annealed approximation. *Europhys. Lett.*, 1:45–52.
- Derrida, B. and Weisbuch, G. (1986). Evolution of overlaps between configurations in random boolean networks. *J. Physique*, 47:1297.
- Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Perseus Books.
- Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. Technical report, German National Research Center for Information Technology. GMD Report 159.
- Kauffman, S. A. (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press.
- Langton, C. G. (1990). Computation at the edge of chaos. *Physica D*, 42.
- Luque, B. and Sole, R. (2000). Lyapunov exponents in random boolean networks. *Physica A*, 284:33–45.
- Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560.

- Mitchell, M., Hraber, P. T., and Crutchfield, J. P. (1993). Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130.
- Natschläger, T. and Maass, W. (2003). Information dynamics and emergent computation in recurrent circuits of spiking neurons. In *Proc. of NIPS 2003, Advances in Neural Information Processing Systems*, volume 16. MIT Press. to appear.
- Packard, N. (1988). Adaptation towards the edge of chaos. In Kelso, J. A. S., Mandell, A. J., and Shlesinger, M. F., editors, *Dynamic Patterns in Complex Systems*, pages 293–301. World Scientific.
- Rohlf, T. and Bornholdt, S. (2002). Criticality in random threshold networks: annealed approximation and beyond. *Physica A*, 310:245–259.
- Siu, K.-Y., Roychowdhury, V., and Kailath, T. (1995). *Discrete neural computation; a theoretical foundation*. information and system sciences series. Prentice-Hal.
- van Vreeswijk., C. A. and Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274:1724–1726.

A Time evolution of Hamming distance $d(t)$

To determine the time evolution of the Hamming distance we consider a randomly drawn set of weights w_{ij} and the following scenario:

- 1) The state $\mathbf{x}_1(t)$ is mapped via Equ. (1) to the state $\mathbf{x}_1(t + 1)$ where u_1 is given as input.
- 2) The state $\mathbf{x}_2(t)$ is mapped by the *same* network to the state $\mathbf{x}_2(t + 1)$ where u_2 is given as input.

We will derive an equation which relates the Hamming distance $d(t)$ at time t between the two states $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ to the Hamming distance $d(t + 1; u_1, u_2)$ at time t between the two states $\mathbf{x}_1(t + 1)$ and $\mathbf{x}_2(t + 1)$ given the two inputs u_1 and u_2 .

Assume that out of the K inputs $x_j(t)$ to a given unit i there are exactly c ($0 \leq c \leq K$) which are different between the states $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$, i.e. $c = |\mathcal{D}|$ with $\mathcal{D} = \{j : x_{1,j}(t) \neq x_{2,j}(t) \wedge w_{ij} \neq 0\}$. The resulting internal activations at time $t + 1$ of a unit i in the two cases 1) and 2) can be written as (index i and t omitted for brevity)

$$\begin{aligned} h_1 &= \sum_j w_{ij} x_{1,j} + u_1 = a + b + u_1 \\ h_2 &= \sum_j w_{ij} x_{2,j} + u_2 = a - b + u_2 \end{aligned}$$

where $a = \sum_{j \in \mathcal{E}} w_{ij}$ is a sum over the $K - c$ state components that are equal ($\mathcal{E} = \{j : x_{1,j}(t) = x_{2,j}(t) \wedge w_{ij} \neq 0\}$) and $b = \sum_{j \in \mathcal{D}} w_{ij}$ is the sum over the c differing state components ($\mathcal{D} = \{j : x_{1,j}(t) \neq x_{2,j}(t) \wedge w_{ij} \neq 0\}$) (cf. (Derrida, 1987)).

In the following we apply the annealing approximation introduced by Derrida and others (Derrida and Pomeau, 1986). The basic assumption of the annealing approximation in our context is that at each time step the whole weight matrix is randomly generated. This has the effect that correlations between units which would otherwise be built up during previous time steps must not be considered. Although this is a rather radical assumption the predictions obtained for the quenched case (weight matrix generated once) are quite accurate; see Fig. 2. This is due to the fact that if the number of inputs to a unit is rather small compared to the network size ($K \ll \ln N$) then correlations between units build up very slowly and can be neglected in the limit $N \rightarrow \infty$ (Derrida, 1987; Derrida and Weisbuch, 1986).

We calculate the probability $P_{BF}(c, u_1, u_2)$ that the states $x_{1,i}(t+1)$ and $x_{2,i}(t+1)$ of a node i at time $t + 1$ differ given that $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ differ in exactly c bits and u_1 and u_2 are the external inputs to the nodes independently of i . Note that $P_{BF}(c, u_1, u_2)$ is equal to the probability that h_1 and h_2 have different signs.

$$\begin{aligned} P_{BF}(c, u_1, u_2) &= \Pr \{x_{1,i}(t+1) \neq x_{2,i}(t+1)\} \\ &= \Pr \{\Theta(h_1) \neq \Theta(h_2)\} \\ &= \Pr \{h_1 \geq 0 \wedge h_2 < 0\} + \Pr \{h_1 < 0 \wedge h_2 \geq 0\} \\ &= \Pr \{a \geq -b - u_1 \wedge a < b - u_2\} + \Pr \{a < -b - u_1 \wedge a \geq b - u_2\} \end{aligned} \tag{8}$$

Under the approximations made a Hamming distance of $d(t)$ can be interpreted as the probability that a certain component of two states $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ differs.

Then the probability that out of the K inputs to a given node i exactly c differ is given by the binomial distribution $\binom{K}{c} \cdot d(t)^c \cdot (1 - d(t))^{K-c}$. Averaging over all possible values of c we arrive at the following formula relating the Hamming distance in one time-step to the next:

$$d(t + 1; u_1, u_2) = \sum_{c=0}^K \binom{K}{c} \cdot d(t)^c \cdot (1 - d(t))^{K-c} \cdot P_{BF}(c, u_1, u_2) \quad (9)$$

For weights drawn from a Gaussian distribution with zero mean a and b are normally distributed with zero mean and variances $\sigma_a^2 = (K - c)\sigma^2$ and $\sigma_b^2 = c\sigma^2$ respectively and P_{BF} can be calculated as the integral over the area in the (a, b) plane corresponding to the constraints defined in Equ. (8).

$$P_{BF}(c, u_1, u_2) = \int_{-\infty}^{\infty} \left(\phi(b, 0, c\sigma^2) \int_{\min\{b-u_2, -b-u_1\}}^{\max\{b-u_2, -b-u_1\}} \phi(a, 0, (K-c)\sigma^2) da \right) db \quad (10)$$

for $0 < c < K$ where $\phi(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2} \cdot \frac{(x-\mu)^2}{\sigma^2}}$ denotes the Gaussian density function. Note that in the cases of $c = 0$ and $c = K$ the equation for $P_{BF}(c, u_1, u_2)$ collapses to a single integral:

$$P_{BF}(0, u_1, u_2) = \int_{\min\{-u_2, -u_1\}}^{\max\{-u_2, -u_1\}} \phi(a, 0, K\sigma^2) da$$

$$P_{BF}(K, u_1, u_2) = 1 - \int_{\min\{-u_1, u_2\}}^{\max\{-u_1, u_2\}} \phi(b, 0, K\sigma^2) db$$

This can easily be concluded by considering Equ. (8) and noting that $c = 0 \Rightarrow b = 0$ and $c = K \Rightarrow a = 0$.

Note that we use the shorthands $d(t, u) := d(t, u, u)$ and $P_{BF}(c, u) := P_{BF}(c, u, u)$ in the main text.

B Calculation of the criticality criterion

The derivative of $\frac{\partial d_{fade}(t+1, u_1, u_2)}{\partial d}$ at $d = 0$ can be calculated as follows (we write $P(c)$ as shorthand for $P_{BF}(c, u_1, u_2)$):

$$\begin{aligned}
\left(\frac{\partial}{\partial d} \sum_{c=0}^K \binom{K}{c} d^c (1-d)^{K-c} P(c) \right) \Big|_{d=0} &= \left(\frac{\partial}{\partial d} \binom{K}{0} (1-d)^K P(0) \right) \Big|_{d=0} \\
&+ \left(\frac{\partial}{\partial d} \binom{K}{1} d (1-d)^{K-1} P(1) \right) \Big|_{d=0} \\
&+ \left(\sum_{c=2}^K \binom{K}{c} \frac{\partial}{\partial d} d^c (1-d)^{K-c} P(c) \right) \Big|_{d=0} \\
&= (K(-1)(1-d)^{K-1} P(0)) \Big|_{d=0} \\
&+ (K((1-d)^{K-1} - d(K-1)(1-d)^{K-2}) \\
&\quad P(1)) \Big|_{d=0} \\
&+ 0 \\
&= K(P(1) - P(0))
\end{aligned}$$

Since the dynamic of the network is deterministic the same input never leads to an output bit-flip, i.e. $P_{BF}(0, u, u) = 0$, and $d_{fade}(t+1) = r \cdot d(t+1; \bar{u}+1, \bar{u}+1) + (1-r) \cdot d(t+1; \bar{u}-1, \bar{u}-1)$ we obtain:

$$\frac{\partial d_{fade}(t+1)}{\partial d(t)} \Big|_{d(t)=0} = K \cdot (r \cdot P_{BF}(1, \bar{u}+1, \bar{u}+1) + (1-r) \cdot P_{BF}(1, \bar{u}-1, \bar{u}-1))$$

Investigation of the special cases $c = 1, K = 1, u_1 = u_2 = u$ and $c = 1, K = 2, u_1 = u_2 = u$ will show that for $K = 1$ and $K = 2$ the network is always in the ordered phase regardless of the parameters σ^2 and \bar{u} .

For $c = 1, K = 1$ and $u_1 = u_2 = u$ we get

$$P_{BF}(1, u, u) = 1 - \int_{\min\{-u, u\}}^{\max\{-u, u\}} \phi(b, 0, \sigma^2) db < 1$$

Hence for the criticality criterion Equ. (4) we obtain

$$r \cdot \underbrace{P_{BF}(1, \bar{u}+1, \bar{u}+1)}_{<1} + (1-r) \cdot \underbrace{P_{BF}(1, \bar{u}-1, \bar{u}-1)}_{<1} < \frac{1}{K} = 1$$

since $0 \leq r \leq 1$.

For $c = 1, K = 2$ and we get

$$P_{BF}(1, u, u) = \int_{-\infty}^{\infty} db \phi(b, 0, \sigma^2) \int_{\min\{b-u, -b-u\}}^{\max\{b-u, -b-u\}} da \phi(a, 0, \sigma^2)$$

Note that this is an integral over the product density $\phi(b, 0, \sigma^2) \cdot \phi(a, 0, \sigma^2)$ with equal variance in each dimension. Taking into account the area of integration which is spanned by the two lines $b - u$ and $-b - u$ and covers half of the (a, b) plane it is clear that at most half of the probability mass (in the case of $u = 0$) is covered by that integral, i.e. $P_{BF}(1, u, u) < 0.5$ for $u \neq 0$ and $P_{BF}(1, 0, 0) = 0.5$. Hence the criticality criterion reads as follows

$$r \cdot \underbrace{P_{BF}(1, \bar{u} + 1, \bar{u} + 1)}_{<0.5} + (1 - r) \cdot \underbrace{P_{BF}(1, \bar{u} - 1, \bar{u} - 1)}_{<0.5} < \frac{1}{K} = 0.5$$

since $0 \leq r \leq 1$.

C Correction term for the NM -separation

We first calculate the fraction $q(\bar{u}, b)$ of neurons which copy the input. A unit i copies the input $u(t) = \bar{u} + \beta(t)$ with the bit $\beta \in \{-1, +1\}$ if the output has the same value as $\beta(t)$, i.e. $x_i(t) = \beta(t)$. $q(\bar{u}, r)$ is given as follows

$$\begin{aligned} q(\bar{u}, r) := & r \cdot \Pr \left\{ \sum_j w_{ij} x_j(t-1) + \bar{u} + 1 \geq 0 \right\} \\ & + (1 - r) \cdot \Pr \left\{ \sum_j w_{ij} x_j(t-1) + \bar{u} - 1 < 0 \right\} \end{aligned}$$

where $r = \Pr \{u(t) = \bar{u} + 1\}$. The sum $\sum_j w_{ij} x_j(t-1)$ is normally distributed with zero mean and variance $K\sigma^2$ which yields

$$q(\bar{u}, r) = r \cdot \Phi(\bar{u} + 1, 0, K\sigma^2) + (1 - r) \cdot \Phi(-\bar{u} + 1, 0, K\sigma^2)$$

where $\Phi(x, \mu, \sigma^2)$ denotes the Gaussian cumulative density $\int_{-\infty}^x \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2} \cdot \frac{(z-\mu)^2}{\sigma^2}} dz$.

As the next step we calculate the probability that for two given inputs u_1 and u_2 with $b := \Pr \{u_2 \neq u_1\}$ the outputs $x_1 = \Theta(\sum_j w_{ij}x_j(t-1) + u_1)$ and $x_2 = \Theta(\sum_j w_{ij}x_j(t-1) + u_2)$ of a unit i are different. $\Pr \{x_1 \neq x_2\}$ can be interpreted as the Hamming distance which is generated immediately by the input.

$$\begin{aligned} \Pr \{x_1 \neq x_2\} = & \Pr \{u_1 = u_2\} \cdot (\Pr \{u_1 \text{ is copied and } u_2 \text{ is not copied}\} + \\ & \Pr \{u_1 \text{ is not copied and } u_2 \text{ is copied}\}) + \\ & \Pr \{u_1 \neq u_2\} \cdot (\Pr \{u_1 \text{ is copied and } u_2 \text{ is copied}\} + \\ & \Pr \{u_1 \text{ is not copied and } u_2 \text{ is not copied}\}) \end{aligned}$$

This can be simplified to (arguments (\bar{u}, r) dropped for sake of brevity)

$$\Pr \{x_1 \neq x_2\} = 2(1-b)q(1-q) + b(q^2 + (1-q)^2) .$$

Since we want to get an estimate of the Hamming distance due to the difference b of the input we correct the above term by $\Pr \{x_1 \neq x_2\} |_{b=0}$ and use the result as the correction term introduced in section 4.

$$\begin{aligned} \Pr \{x_1 \neq x_2\} - \Pr \{x_1 \neq x_2\} |_{b=0} &= 2(1-b)q(1-q) + b(q^2 + (1-q)^2) - 2q(1-q) \\ &= b(2q-1)^2 \end{aligned}$$